Trabalho 3

Introdução ao Processamento Digital de Imagem (MC920/MO443)
Professor: Hélio Pedrini

Vinicius Couto Espindola | RA: 188115

17 de Maio de 2020

1 Introdução

A transformada de Fourier consiste em, essencialmente, representar a mesma informação de uma forma diferente. Em processamento de imagens, a transformada representa a imagem em diversas componentes de frequência, as quais, quando combinadas adequadamente, compõem a mesma imagem que a original. Como estamos tratando imagens, temos que utilizar a transformada de Fourier para duas dimensões. Este processo consiste em aplicar a transformada em todas as colunas e depois sobre as linhas resultantes, gerando, assim, uma matriz de coeficientes de Fourier. Com esta representação da imagem no domínio da frequência, temos uma série de operações que podem ser aplicadas de forma engenhosa. Neste trabalho, focaremos em como utilizar a *Fast Fourier Transform* (FFT) de imagens para aplicação de filtros e algoritimos de compressão.

2 Execução

O programa requer as seguintes biblotecas: *OpenCV*, *NumPy* e *Matplotlib*. Sua execução pode tanto exibir a imagem de saida, quanto salvar o resultado em um arquivo. Todo os resultados produzidos e utilizados neste resultado encontram-se na pasta "*./outputs*". As imagens de entrada utilizadas estão na pasta "*./inputs*".

$$assignment\_3.py\ [-h]\ input\ \{-f\ filter\ |\ -c\ percentage\}\ [-r1\ float]\ [-r2\ float]\ [-p]\ [-o]$$

input: *Path* da imagem de entrada a ser processada.
-f filter: Deve ser uma string dentre *"low", "band" e "high"* correspondendo ao filtro que se deseja aplicar.
Nota: Caso *-f* seja usado, *-c* não deve ser incluido como parâmetro. Caso seja selecionado o filtro "*band*", deve-se definir tando *r1* quando "r2", caso seja algum dos outros, apenas "*r1*" é definido.
-r1 e -r2: São os raios utilizados para o processo de filtragem. Cada um destes recebe um float como parâmetro.
-p ou –pipeline: Esta flag permite ver o estado da imagem durante todo o processo de aplicação dos filtros. Será gerado um plot contendo cada uma das imagens de cada etapa.
-c percentage: Esta opção seleciona a compressão de imagem. Deve-se passar uma porcentagem (float positivo de 0 à 100) a qual será utilizar para definir o limiar de truncamento das frequências.
-o output-file: *Path* onde a imagem de saída deve ser salva.
Nota: Caso não seja definido um *output-file*, a imagem será exibida na tela. Se for definido, a imagem de saída será salva apenas.

3 Soluções

A biblioteca `Numpy` já apresenta funções prontas para se calcular a FFT 2D de matrizes, também apresenta funções para se realizar a cetralização da transformada. Utilizou-se, então, a função `np.fft.fft2(img)`

para calcucar a FFT das imagens e a `np.fft.fftshift(transformed)` para centralizar a frequência zero da transformada na matriz resultante (melhora a visualização). Para realizar o processo de volta, utilizou-se as inversas destas funções: `np.fft.ifft2(img)` e `np.fft.ifftshift(transformed)`. Para analizar a transformada no espectro de magnitude, utilizou-se o seguinte operador logarímitico: $Q(i,j) = c \cdot log(1 + |P(i,j)|)$. O cáculo de matrizes para os filtros foram feitos com o simples cálculo das distâncias euclidianas entre as células da matriz e o centro da mesma.

3.1 Filtros Utilizando FFT

A matriz resultante da transformada feita a partir das funções do `Numpy`, representa a mesma imagem mas como uma somatória de senos/cossenos. Observa-se que imagens com padrões simples vinculados às funções seno e cosseno apresentam transformadas igualmente simples. A visualização da FFT, também, requer uma transformação dos conjuntos de números complexos para, por exemplo, a representação da magnitude ou da fase das frequências. A magnitude foi escolhida por representar melhor as propriedades geométricas da imagem original, ainda assim, o cáculo da magnitude pode ser feito de diferente formas (será utilizando o operador logarítimico já descrito).

A transformada possibilita aplicações de filtros de imagens de uma maneira intrigante: após o deslocamento do da frequência zero para o centro, temos que as baixas frequências são concentradas próximas ao centro da matriz, enquanto as altas são mantidas nas regiões marginais. Assim, para realizar a aplicação de filtros, podemos simplesmente zerar as coordenadas que estão mais próximas do centro (passa-altas) ou as mais externas (passa-baixas). Também podemos combinar estes filtros para criar um filtro que aceite apenas frequências de uma determinada faixa (passa-bandas).

Para aplicar tais filtros, então, definimos um (ou dois) raio(s) que será(ão) usados para traçar a fronteira entre as frequências inalteradas e as zeradas. Para o filtro passa-baixas, o raio determina a área que terá as frequências inalteradas, no passa-altas, a área das frequências que serão zeradas. No passa-bandas, serão definidos dois raios que delimitaram a coroa na qual as frequências ficarão inalteradas. Os filtros são aplicados da seguinte maneira: a imagem primeiro é transformada, em seguida, é filtrada, e, então, aplica-se a transformada inversa para obter a imagem resultante. A Figura 1 ilustra os passos do processo.

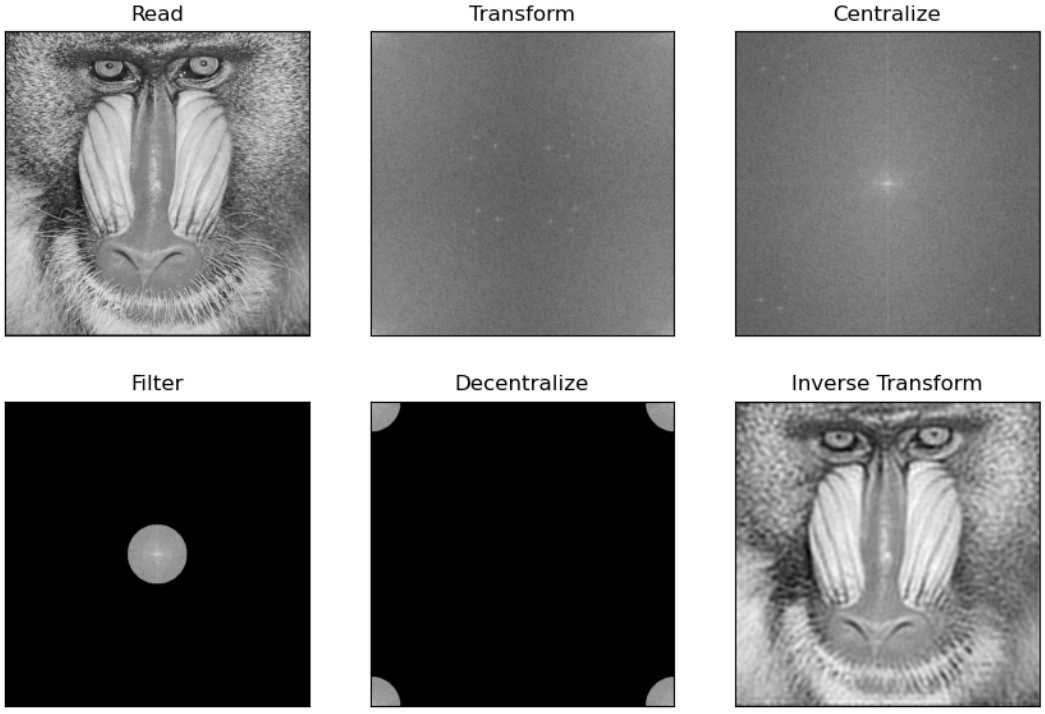

Figure 1: Pipeline da aplicação de filtros usando FFT.

3.1.1 Filtro Passa-Baixas

O filtro passa-baixas se comporta como esperado. É notável que, mantendo um número pequeno das componentes (o caso **a** da Figura 2 mantém menos de 50% das componentes), preserva-se maior parte da informação da imagem. Esta fato serve como motivação para realizar a compressão das imagens utilizado FFT, uma vez que mantendo tão poucos dados da imagem, conseguimos preserva-lá tão bem. Por volta de 50 pixels de raio, temos que o passa-baixas passa a atuar como um filtro suavizador da imagem, chegando a borrá-la quase inteiramente a partir dos 10 pixels de raio.

(a) $r1 = 100px$ (b) $r1 = 50px$ (c) $r1 = 10px$

Figure 2: Resultados gerados pelo filtro **Passa-Baixas**.

3.1.2 Filtro Passa-Altas

Para o filtro passa-altas, deixar um raio pequeno gera uma imagem próxima à original. Na figura 3 podemos ver que a casa tem suas bordas delimitadas, mas regiões como as nuvens ainda são visíveis. Aumentando o raio, aumentamos a área filtrada e consequentemente, as regiões brancas homogêneas desaperecem, permanecendo apenas as bordas mais significativas como o contorno da casa e das árvores. Diferentemente do passa-baixas, o passa-altas mostrou-se extremamente sensível. Como podemos observar na Figura 3, mesmo filtrando apenas áreas minúsculas (o caso **a**, por exemplo, tem uma área menor que $3px^2$) a imagem já apresenta alterações significativas. Isso decorre do fato que estamos filtrando frequências que são componentes significativas da imagem (possuem grande magnitude).

(a) $r1 = 3px$ (b) $r1 = 10px$ (c) $r1 = 30px$

Figure 3: Resultados gerados pelos filtros **Passa-Altas**.

3.1.3 Filtro Passa-Bandas

Os resultados apresentados pelo filtro passa-bandas, exibidos na Figura 4, não apresentam muita diferença quando comparados aos outros dois filtros já analisados. O passa bandas nada mais é que uma somatória entre um passa-baixa e um passa-alta. Assim, conseguimos manter um controle mais preciso da faixas de frequências que são permeadas pelo filtro. No caso **a** da Figura 4, temos um raio interno pequeno e um externo grande, isso faz com que a imagem se aproxime da original, mas, como visto no passa altas, as baixas frequências formam parte significativa da imagem (apresentam grande amplitude), então, mesmo com um raio interno de 2px, a imagem perde uma quantia significativa de informação. Para bandas intermediárias como é o caso **b**, temos um efeito de marcação de boradas (passa-altas) mas estas se apresentam suavizadas (passa-baixa), o que exemplifica a combinação de efeitos que o passa-bandas gera. No caso **c**, a faixa já é muito estreita e grande parte da informação da imagem é perdida.

(a) $r1 = 2px$ & $r2 = 300$ (b) $r1 = 10px$ & $r2 = 40$ (c) $r1 = 30px$ & $r2 = 100$

Figure 4: Resultados gerados pelo filtro **Passa-Bandas**.

3.2 Compressão Utilizando FFT

Mesmo que pequenas, imagens digitais apresentam um vasto número de possíveis representações. Para uma imagem preto e branco de $20x20$ pixels, temos da ordem de 2^{400} imagens representáveis. No entanto, muitas destas possiblidades são irrelevantes por não apresentarem algum padrão significativo (não seriam pessoas ou animais, mas apenas valores aleatórios). A compressão visa reduzir o espaço de possíveis imagens representadas de forma a reduzir o total de imagens representáveis afetando o mínimo possível as representações que possuem algum padrão significativo.

Aplicando a FFT sobre imagens, nota-se que muitas destas são frequências de pequena magnitude, ou seja, refletem pouco a composição final da imagem. Com isso, podemos aplicar uma técnica de compressão na FFT de uma imagem onde truncamos para zero as frequências de magnitude abaixo de um limiar. No caso, o limiar será definido de acordo com uma porcentagem do intervalo da menor à maior magnitude das frequências da FFT ($limiar = min + (max - min) \cdot porcentagem$), esta porcentagem pode ser passada como parâmetro para o programa. Também foi feita a contagem de quantas células são zeradas ao aplicar a compressão, assim podemos ter uma idéia de o quão bom é a compressão em questão.

(a) 25% - 6903 truncamentos (b) 40% - 144341 truncamentos (c) 45% - 227183 truncamentos

(d) 50% - 253899 truncamentos (e) 55% - 260335 truncamentos (f) 70% - 262077 truncamentos

Figure 5: Resultados gerados pela compressão FFT usando o limiar percentual.

Na Figura 5, observa-se que a compressão pode ser extremamente eficiente. Para o caso **c**, temos 227183 valores truncados para zero, sendo que a imagem possui 262144 valores no total, ou seja, cerca de 87% das componentes são descartadas. Ainda por cima, o caso **c** retem uma boa quatia de informação da imagem original. Nota-se, também, que compressões mais extremas, como são os casos **d**, **e** e **f**, não são muito benéficas, já que apresentam um aumento pequeno do número de truncamentos para perdas siginificativas na qualidade de imagens quando comparados com o caso **c**.

Uma observação interesante do caso **f** da Figura **5**, é que, como muitas componentes são removidas da FFT, temos menos inteferência entre as frequências, consequentemente, conseguimos ver traços horizontais e verticais na imagem. Estes traços correspondem as principais componentes seno e cosseno que formam a imagem original.

Nota-se que para imagens com mais variações no domínio de frequência tivemos resultados melhores, obtendo imagens com várias células truncadas e com pouca perda de informação. No entanto, imagens com mais superfícies homogêneas e menos bordas, nota-se que há perdas significativas na imagem para compressões menores.

3.3 Limitações

O processo utilizado é bem robusto quanto à limitações. A principal restrição é apenas imagens em escala de cinza são processadas. Aindas assim, qualquer imagem, idependente de tamanho, deve funcionar com o programa criado.

4 Discussão e Conclusão

Assim como em outras inúmeras áreas do conhecimento, a transformada de Fourier apresenta-se como uma ferramenta poderosa para o processamento digital de imagens. A possibilidade de manupular a imagem a partir de um representação fundada em uma característica da imagem, possibilida a aplicação de processos já conhecidos de maneira eficiente. Em particular, a compressão se mostrou extremamente eficaz para algumas imagens, conseguindo eliminar mais da metade das componentes que representam a imagem sem que houvesse grande perdas na qualidade da imagem. Os filtros aplicados também apresentaram-se úteis devido a simplicidade que estes proporcionam: não precisamos criar núcleos com constantes e aplicar convolução sobre a imagem. Ao usarmos a FFT, podemos criar filtros baseados em um único valor, o que facilita a implementação e regulação de parâmetros. Aqui tratamos apenas duas das aplicações que a FFT traz para imagens, todavia há inúmeras outras utilidades para esta no campo de processamento digital de imagem.

References

[1] R.C. Gonzalez. *Digital Image Processing*. Prentice Hall, 2007.